Data structures and Algorithms

Superset ID- 6363229

**Exercise 2: E-commerce Platform Search Function**

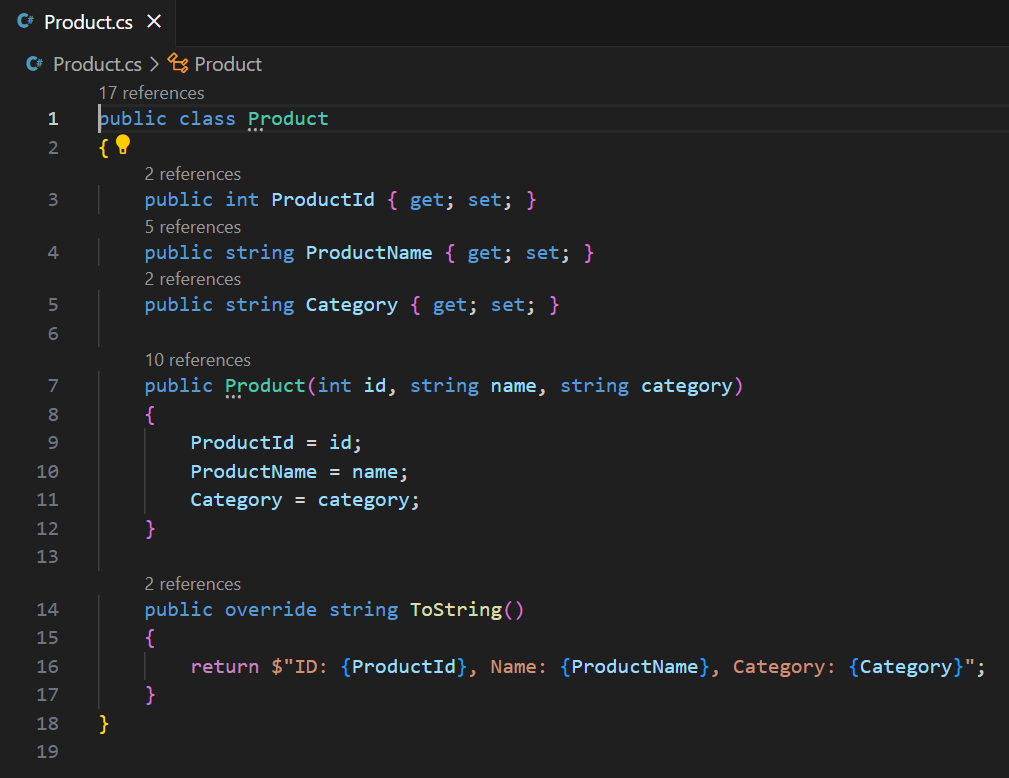
**Understand Asymptotic Notation:**

Big O notation describes how the time (or space) complexity of an algorithm scales as the input size increases.

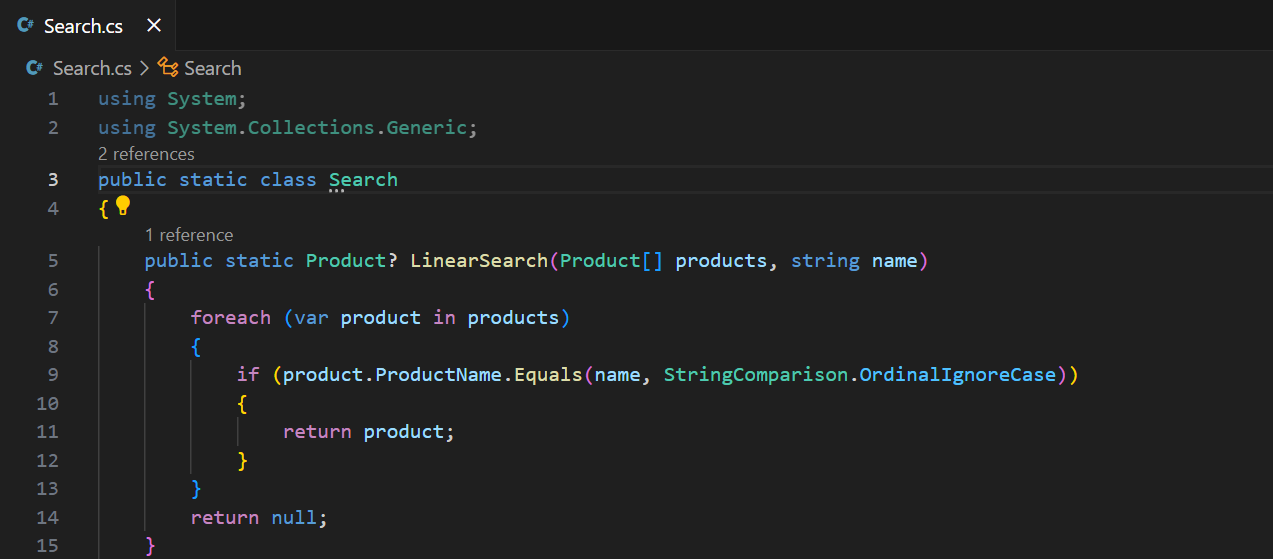
| **Case** | **Description** |
| --- | --- |
| **Best Case** | The fastest scenario (e.g., item is first) |
| **Average Case** | Expected time for typical inputs |
| **Worst Case** | The slowest scenario (e.g., item not found) |

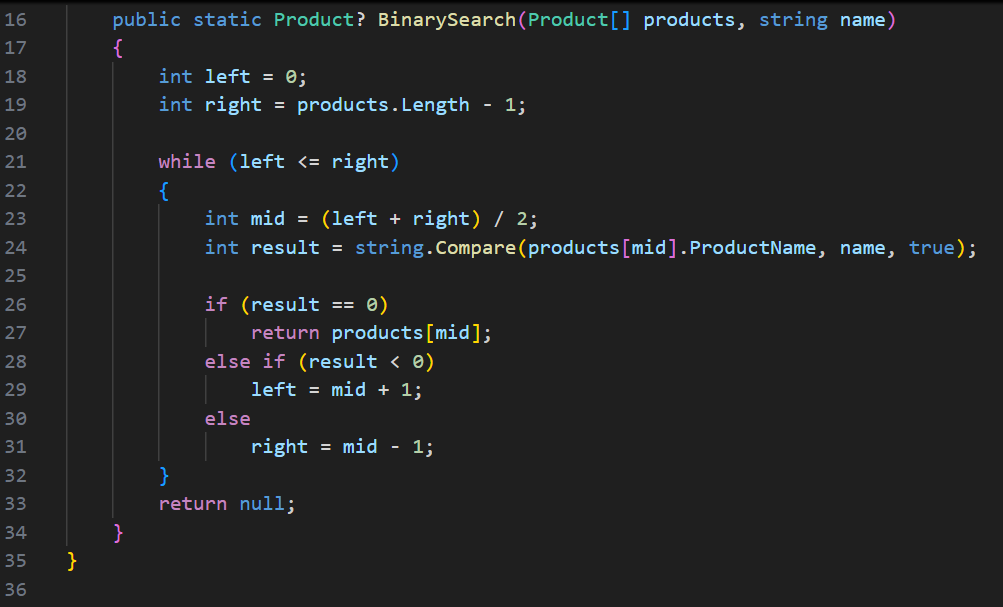
Implementation:

Product.cs

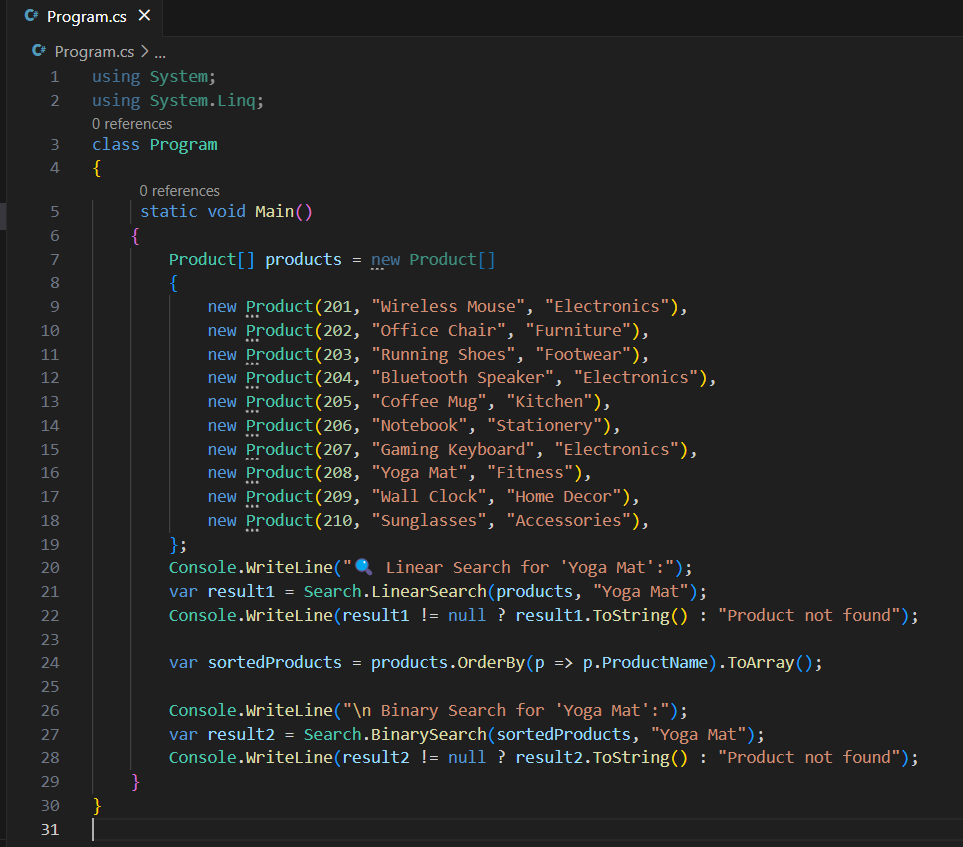


Search.cs

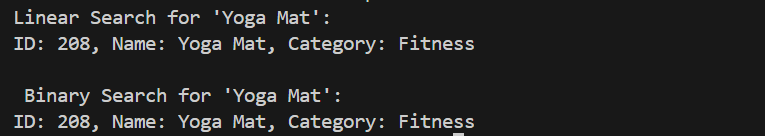




Program.cs



Output:



**Analysis:**

**Linear Search**

1. Simple to implement, no sorting required.
2. Inefficient for large datasets — scans every element.

**Binary Search**

1. Much faster for large datasets (O(log n)).
2. Requires sorted data.
3. Ideal for high-traffic e-commerce platforms where performance matters.

For your e-commerce platform, use:

Binary Search (with sorting and indexing) for fast product lookup.Or consider advanced search systems like:

Hash tables for exact matches (O(1) avg).

Search engines (e.g., Elasticsearch) for fuzzy/full-text search at scale.

**Exercise 7: Financial Forecasting**

**Understand Recursive Algorithms**

Recursion is a technique where a method calls itself to solve smaller subproblems. It is often used when a problem can be broken down into simpler instances of itself.

**Predicting Future Value**

We’ll use a simple financial forecasting formula using compound growth:

FV=PV×(1+r)n

Where:

FV = Future Value

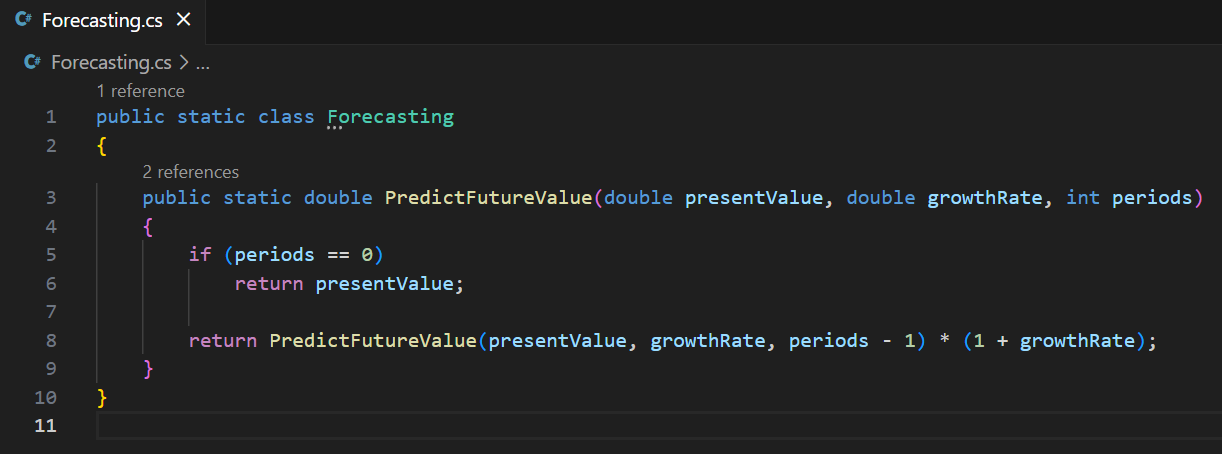
PV = Present Value

r = Growth rate per period

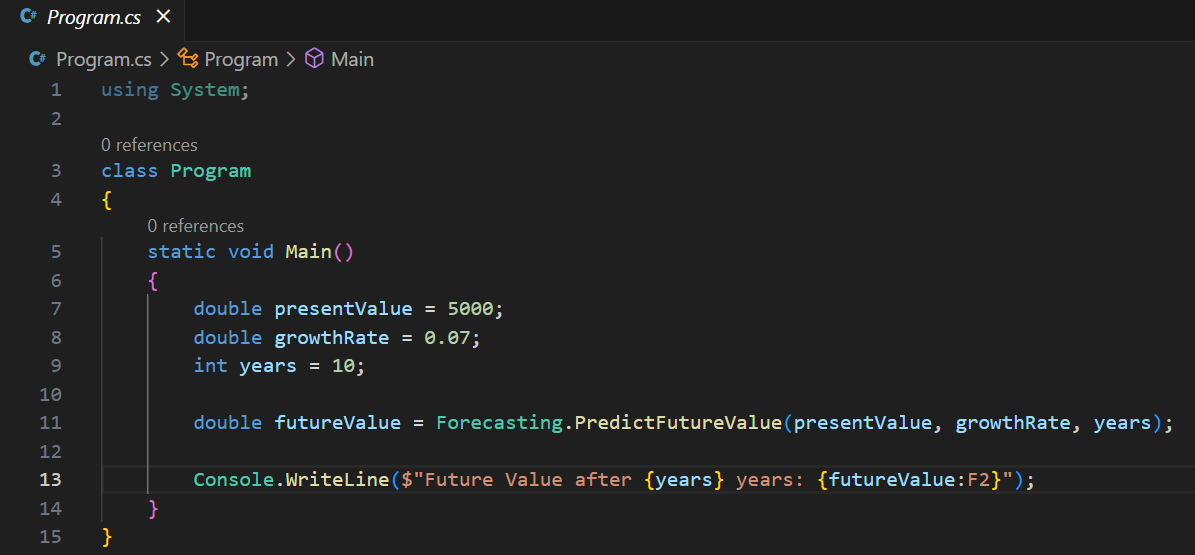
n = Number of periods

**Implementation:**

Forecasting.cs



Program.cs



Output:



**Analysis:**

**Time Complexity**

1. The recursive function has O(n) time complexity, since it makes n recursive calls (where n is the number of periods).
2. Each call performs a simple multiplication.

**Problem with Recursion**

1. For large n, recursion may lead to stack overflow or excessive overhead.
2. It's not tail-recursive, so each call adds a new frame to the call stack.

**Optimization:**

Instead of relying on recursion, you can convert the logic into a loop-based (iterative) version, which:

1. Avoids stack usage: Loops don’t create new function frames.
2. Improves performance: Iteration is generally faster and more memory-efficient.
3. Easier to debug and scale: Especially important for financial systems that need to forecast hundreds or thousands of periods.